APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/783,002	02/20/2004	Fuhwei Lwo	RSW920030291US1	7441

45541   7590   04/16/2009
HOFFMAN WARNICK LLC
75 STATE ST
14TH FLOOR
ALBANY, NY 12207

EXAMINER
CHEN, QING

ART UNIT	PAPER NUMBER
2191	

NOTIFICATION DATE	DELIVERY MODE
04/16/2009	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

PTOCommunications@hoffmanwarnick.com

	Application No.	Applicant(s)	
Office Action Summary	10/783,002	LWO, FUHWEI	
	Examiner	Art Unit	
	Qing Chen	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE <u>3</u> MONTH(S) OR THIRTY (30) DAYS,
WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.
- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed
  after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
  Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any
  earned patent term adjustment. See 37 CFR 1.704(b).

Status

1) ☒ Responsive to communication(s) filed on <u>21 January 2009</u>.

2a) ☒ This action is **FINAL.**   2b) ☐ This action is non-final.

3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is
closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4) ☒ Claim(s) <u>1-27</u> is/are pending in the application.

    4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5) ☐ Claim(s) _____ is/are allowed.

6) ☒ Claim(s) <u>1-27</u> is/are rejected.

7) ☐ Claim(s) _____ is/are objected to.

8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9) ☒ The specification is objected to by the Examiner.

10) ☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

    Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

    Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

    a)☐ All   b)☐ Some * c)☐ None of:

      1.☐ Certified copies of the priority documents have been received.

      2.☐ Certified copies of the priority documents have been received in Application No. _____.

      3.☐ Copies of the certified copies of the priority documents have been received in this National Stage
         application from the International Bureau (PCT Rule 17.2(a)).

    * See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☐ Information Disclosure Statement(s) (PTO/SB/08)
   Paper No(s)/Mail Date _____.

4) ☐ Interview Summary (PTO-413)
   Paper No(s)/Mail Date. _____

5) ☐ Notice of Informal Patent Application

6) ☐ Other: _____.

DETAILED ACTION

1.   This Office action is in response to the amendment filed on January 21, 2009.

2.   **Claims 1-27** are pending.

3.   **Claims 1, 4-6, 9, 10, 12-15, 18, 19, 21-24, and 27** have been amended.

4.   The objection to the abstract is maintained in view of Applicant's amendments to the

abstract and further explained hereinafter.

5.   The objection to the specification is withdrawn in view of Applicant's amendments to the

specification.

6.   The 35 U.S.C. § 112, second paragraph, rejections of Claims 1, 4-6, 10, 12-15, 18, 19,

21-24, and 27 are withdrawn in view of Applicant's amendments to the claims. However,

Applicant's amendments to the claims fail to fully address the rejections of Claims 3 and 9 due

to the use of trademarks. Accordingly, these rejections are maintained and further explained

hereinafter.

Response to Amendment

Specification

7.   The abstract of the disclosure is objected to because the heading of the abstract should be

"Abstract" or "Abstract of the Disclosure." See 37 CFR § 1.72(b).

Claim Rejections - 35 USC § 112

8.   The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the
> subject matter which the applicant regards as his invention.

9.   **Claims 3-6 and 9** are rejected under 35 U.S.C. 112, second paragraph, as being indefinite

for failing to particularly point out and distinctly claim the subject matter which applicant

regards as the invention.

        Claims 3 and 9 contain the trademark or trade name JAVA. When a trademark or trade

name is used in a claim as a limitation to identify or describe a particular material or product, the

claim does not comply with the requirements of the 35 U.S.C. 112, second paragraph. *Ex parte*

Simpson, 218 USPQ 1020 (Bd. App. 1982). The claim scope is uncertain since the trademark or

trade name cannot be used properly to identify any particular material or product. A trademark or

trade name is used to identify a source of goods, and not the goods themselves. Thus, the use of a

trademark or trade name in a claim to identify or describe a material or product (in the present

case, a specific programming language) would not only render a claim indefinite, but would also

constitute an improper use of the trademark or trade name.

        Claims 4-6 depend on Claim 3 and, therefore, suffer the same deficiency as Claim 3.

                        Claim Rejections - 35 USC § 103

10.   The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all

obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in
> section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are
> such that the subject matter as a whole would have been obvious at the time the invention was made to a person
> having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the
> manner in which the invention was made.

11.   **Claims 1-6, 8, 10-15, 17, 19-24, and 26** are rejected under 35 U.S.C. 103(a) as being

unpatentable over **US 6,986,132 (hereinafter "Schwabe")** in view of **US 6,430,564**

(hereinafter "Judge").

As per **Claim 1**, Schwabe discloses:

-   receiving source input corresponding to a first release of byte code and target input

corresponding to a second release of the byte code *(see Figure 20A: 1540 and 1550)*;

-   transforming the source input into a first list that contains class names associated with

the first release of byte code, and the target input into a second list containing class names

associated with the second release of the byte code *(see Figure 20A: 1535 and 1545; Column 14:*

14-15, "An API definition file defines the context of a binary file in relationship to other

referenced binary files.");

-   finding matching class names between the first list and the second list, and loading

classes corresponding to the matching class names *(see Figure 2; Column 4: 1-10, "In the JVM,*

the loading step retrieves the class file representing the desired class."; Column 5: 23-27,

"When the binary file 60 is referenced by an application executing on a virtual machine 65, a

loader 70 loads the binary file 60. A verifier 75 verifies the binary file 60 at some point prior to

execution by an interpreter 80."; Column 25: 44-48, "If the set of classes and interfaces defined

in the old API definition file is not found in the new API definition file …"); and

-   comparing the loaded classes to identify APIs that have been modified between the

first release of byte code and the second release of the byte code, wherein an API has not been

modified in case that it maintains a same name, parameter order, parameter types and return

types in both the first release of the byte code and the second release of the byte code *(see Figure 2; Column 25: 50-53, "... the class and interface attributes are compared to the attributes of the same class or interface in the new package. The attributes may include the name, flags, number of fields and number of methods."; Column 26: 1-10, "At 1650, for each field in the old package, the attributes are compared to the same field in the new package. The attributes may include the name, flags and type." and "At 1655, for each method in the old package, the attributes are compared to the same method in the new package. The attributes may include the name, flags and signature.").*

However, <u>Schwabe</u> does not disclose:

- removing the matching class names from the first list and the second list after the comparing, wherein any class names remaining in the first list represent APIs that have been removed for the second release of the byte code, and wherein any class names remaining in the second list represent APIs that have been added for the second release of the byte code.

<u>Judge</u> discloses:

- removing the matching class names from the first list and the second list after the comparing, wherein any class names remaining in the first list represent APIs that have been removed for the second release of the byte code, and wherein any class names remaining in the second list represent APIs that have been added for the second release of the byte code *(see Column 4: 63-67 through Column 5: 1-5, "Method unloadDataClass unloads a data class by the name of "dataName" by removing the data class object and all instances of the data class object from the data cache 54. Upon removal of a data class object from the data cache 54, the name of*

the data class "dataName" is also removed from the data class list 47 maintained by Data

Manager 48.").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the

invention was made to incorporate the teaching of <u>Judge</u> into the teaching of <u>Schwabe</u> to include

removing the matching class names from the first list and the second list after the comparing,

wherein any class names remaining in the first list represent APIs that have been removed for the

second release of the byte code, and wherein any class names remaining in the second list

represent APIs that have been added for the second release of the byte code. The modification

would be obvious because one of ordinary skill in the art would be motivated to determine

differences between two data lists.

As per **Claim 2**, the rejection of **Claim 1** is incorporated; and <u>Schwabe</u> further discloses:

-    outputting a report identifying at least one of the APIs that have been modified, the

APIs that have been removed and the APIs that have been added *(see Column 25: 44-67 through*

Column 26: 1-10, "... a verification error is indicated.").

As per **Claim 3**, the rejection of **Claim 1** is incorporated; and <u>Schwabe</u> further discloses:

-    wherein the loading step comprises loading at least one Java™ class of the first

release of byte code and at least one class of the second release of the byte code *(see Column 4:*

1-10, "In the JVM, the loading step retrieves the class file representing the desired class.").

As per **Claim 4**, the rejection of **Claim 3** is incorporated; and <u>Schwabe</u> further discloses:

- listing methods of the at least one class of the first release of byte code in the first list, and listing methods of the at least one class of the second release of the byte code in the second list *(see Column 26: 6-10, "At 1655, for each method in the old package, the attributes are compared to the same method in the new package. The attributes may include the name, flags and signature.")*.

As per **Claim 5**, the rejection of **Claim 4** is incorporated; and <u>Schwabe</u> further discloses:
- wherein the comparing step comprises comparing the methods in the first list to the methods in the second list to identify APIs that have been modified between the first release of byte code and the second release of the byte code *(see Column 25: 50-53, "... the class and interface attributes are compared to the attributes of the same class or interface in the new package. The attributes may include the name, flags, number of fields and number of methods.")*.

As per **Claim 6**, the rejection of **Claim 5** is incorporated; however, <u>Schwabe</u> does not disclose:
- wherein the removing step comprises removing, from the first list and the second list, any methods in the first list that are identical to methods in the second list based on the comparison, wherein any methods remaining in the first list after the removing represent APIs that have been removed for the second release of the byte code, and wherein any methods remaining in the second list after the removing represent APIs that have been added for the second release of the byte code.

<u>Judge</u> discloses:

    -   wherein the removing step comprises removing, from the first list and the second list, any methods in the first list that are identical to methods in the second list based on the comparison, wherein any methods remaining in the first list after the removing represent APIs that have been removed for the second release of the byte code, and wherein any methods remaining in the second list after the removing represent APIs that have been added for the second release of the byte code *(see Column 4: 63-67 through Column 5: 1-5, "Method unloadDataClass unloads a data class by the name of "dataName" by removing the data class object and all instances of the data class object from the data cache 54. Upon removal of a data class object from the data cache 54, the name of the data class "dataName" is also removed from the data class list 47 maintained by Data Manager 48.").*

       Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of <u>Judge</u> into the teaching of <u>Schwabe</u> to include wherein the removing step comprises removing, from the first list and the second list, any methods in the first list that are identical to methods in the second list based on the comparison, wherein any methods remaining in the first list after the removing represent APIs that have been removed for the second release of the byte code, and wherein any methods remaining in the second list after the removing represent APIs that have been added for the second release of the byte code. The modification would be obvious because one of ordinary skill in the art would be motivated to determine differences between two data lists.

       As per **Claim 8**, the rejection of **Claim 1** is incorporated; and <u>Schwabe</u> further discloses:

- wherein the source input and the target input comprise a list of classes *(see Column 22: 56-58, "A library or applet package (herein referred to as a binary file) is received (1460) ...").*

Claims 10-15 and 17 are system claims corresponding to the method claims above (Claims 1-6 and 8) and, therefore, are rejected for the same reasons set forth in the rejections of Claims 1-6 and 8.

Claims 19-24 and 26 are program product claims corresponding to the method claims above (Claims 1-6 and 8) and, therefore, are rejected for the same reasons set forth in the rejections of Claims 1-6 and 8.

12. **Claims 7, 9, 16, 18, 25, and 27** are rejected under 35 U.S.C. 103(a) as being unpatentable over **Schwabe** in view of **Judge** as applied to Claims 1, 10, and 19 above, and further in view of **US 6,385,722 (hereinafter "Connelly")**.

As per **Claim 7**, the rejection of **Claim 1** is incorporated; however, <u>Schwabe</u> and <u>Judge</u> do not disclose:

- wherein the source input and the target input comprise JAR files.

<u>Connelly</u> discloses:

- wherein the source input and the target input comprise JAR files *(see Column 1: 14-17, "Software vendors typically ship their products as a set of shared libraries, such as libraries*

written in the Java™ object-oriented programming language and packaged as a conventional

shared library file called a JAR file.").

   Therefore, it would have been obvious to one of ordinary skill in the art at the time the

invention was made to incorporate the teaching of <u>Connelly</u> into the teaching of <u>Schwabe</u> to

include wherein the source input and the target input comprise JAR files. The modification

would be obvious because one of ordinary skill in the art would be motivated to easily and

efficiently share and use these library files *(see <u>Connelly</u> – Column 1: 17-20).*

   Claims 16 and 25 are rejected for the same reason set forth in the rejection of Claim 7.

   As per **Claim 9**, the rejection of **Claim 1** is incorporated; however, <u>Schwabe</u> and <u>Judge</u>

do not disclose:

   -   inputting class paths common to the first release of Java™ byte code and the second

release of the byte code.

   <u>Connelly</u> discloses:

   -   inputting class paths common to the first release of Java™ byte code and the second

release of the byte code *(see Column 7: 46-48, "... the class java.net.URLClassLoader is used as*

class loader 122 to load classes and resources from a class path of JAR files and directory

URLs.").

   Therefore, it would have been obvious to one of ordinary skill in the art at the time the

invention was made to incorporate the teaching of <u>Connelly</u> into the teaching of <u>Schwabe</u> to

include inputting class paths common to the first release of Java™ byte code and the second

release of the byte code. The modification would be obvious because one of ordinary skill in the

art would be motivated to access the parts of shared libraries *(see Connelly – Column 1: 31-34).*

Claims 18 and 27 are rejected for the same reason set forth in the rejection of Claim 9.

Response to Arguments

13.   Applicant's arguments filed on January 21, 2009 have been fully considered, but they are

not persuasive.

In the Remarks, Applicant argues:

a)   For example, with respect to independent claims 1, 10 and 19, Applicant continues to

submit that the cited references fail to teach or suggest comparing the loaded classes to identify

APIs that have been modified between the first release of Java byte code and the second release

of the Java byte code, wherein an API has not been modified in case that it maintains a same

name, parameter order, parameter types and return types in both the first release of the Java byte

code and the second release of the Java byte code. In its argument to the contrary, the Office

cites a passage of Schwabe that describes comparison of package attributes. Col. 25, lines 50-53;

col. 26, lines 1-10. However, a closer reading of the text surrounding the passages cited by the

Office clearly show that these package attributes are part of the content of API definition files.

See e.g., col. 25, lines 37-43. To this extent, the parameters that are compared are in API

definition files. As such, Schwabe never compares loaded classes, but only API definition files

which are separate from these classes. In contrast, Schwabe expressly teaches that

> ...verification does not continue beyond an API definition file. This differs from
> typical verification methods that continue the verification process into an
> implementation of the API definition file. Col. 14, lines 10-13.

To this extent, Schwabe teaches against the loading of classes based on results from an API

definition file verification in its verification process. Furthermore, neither of the references

teaches or suggests the exact comparison of the claimed invention, including, inter alia, order of

parameters and both parameter types and return types. Accordingly, Applicant requests that the

rejection be withdrawn.

Examiner's response:

a)   Examiner disagrees. With respect to the Applicant's assertion that the cited references

fail to teach or suggest comparing the loaded classes to identify APIs that have been modified

between the first release of Java byte code and the second release of the Java byte code, wherein

an API has not been modified in case that it maintains a same name, parameter order, parameter

types and return types in both the first release of the Java byte code and the second release of the

Java byte code, as previously pointed out in the Non-Final Rejection (mailed on 10/20/2008) and

further clarified hereinafter, the Examiner respectfully submits that Schwabe clearly discloses

"comparing the loaded classes to identify APIs that have been modified between the first release

of byte code and the second release of the byte code, wherein an API has not been modified in

case that it maintains a same name, parameter order, parameter types and return types in both the

first release of the byte code and the second release of the byte code" *(see Figure 2; Column 25:*

50-53, "... the class and interface attributes are compared to the attributes of the same class or

interface in the new package. The attributes may include the name, flags, number of fields and

number of methods."; Column 26: 1-10, "At 1650, for each field in the old package, the

attributes are compared to the same field in the new package. The attributes may include the

name, flags and type." and "At 1655, for each method in the old package, the attributes are

compared to the same method in the new package. The attributes may include the name, flags

and signature."). Attention is drawn to Figure 2 of Schwabe which clearly illustrates that within

the JVM, the loader first loads the class files and then the verifier verifies the loaded class files.

Thus, one of ordinary skill in the art would readily comprehend that the verification process

illustrated in Figure 20D occurs after the loading of the class files to compare the package

attributes of the new API definition file and the old API definition file.

Therefore, for at least the reason set forth above, the rejections made under 35 U.S.C. §

103(a) with respect to Claims 1, 10, and 19 are proper and therefore, maintained.

In the Remarks, Applicant argues:

b)   With further respect to independent claims 1, 10 and 19, Applicant continues to submit

that the cited references fail to teach or suggest finding matching class names between the first

list and the second list, and loading classes corresponding to the matching class names. Rather,

the passages in Schwabe refer to different functions of Schwabe that are completely unrelated.

The first passage of Schwabe cited by the Office mentions comparing the set of classes and

interfaces in an old API definition file with those in a new API definition file. The second

passage references loading of class files. However, the passages in Schwabe cited by the Office

do not teach that the class files that are loaded are class files that are derived from the matching.

In fact, Schwabe does not even indicate that the class files that are loaded are taken from its API

definition files. Rather, as stated elsewhere herein, Schwabe teaches against comparing of loaded

classes, but only compares the definition files. This is in contrast to the claimed invention in

which the classes that are loaded correspond to the matching class names between the first list

and the second list. For the above reasons, the separate comparing and loading of Schwabe does

not teach or suggest the loading based on the matching of the claimed invention. Judge does not

cure this deficiency. Accordingly, Applicant requests that the rejection be withdrawn.

Examiner's response:

b)   Examiner disagrees. Applicant's arguments are not persuasive for at least the following

reasons:

  First, with respect to the Applicant's assertion that the passages in Schwabe refer to

different functions of Schwabe that are completely unrelated, as previously pointed out in the

Non-Final Rejection (mailed on 10/20/2008) and further clarified hereinafter, the Examiner

respectfully submits that, in the drawing figure and passages cited by the Examiner, Schwabe

describes class loading and verification in the JVM, which is well-known to those of ordinary

skill in the art. In the JVM, a loader loads a class file prior to a verifier verifies it. Thus, one of

ordinary skill in the art would readily comprehend that loading of class files occurs first in the

JVM and then, the verifier verifies the loaded class files. Therefore, the loading process and the

verification process are related processes in the JVM.

  Second, with respect to the Applicant's assertion that the passages in Schwabe cited by

the Office do not teach that the class files that are loaded are class files that are derived from the

matching, as previously pointed out in the Non-Final Rejection (mailed on 10/20/2008) and

further clarified hereinafter, the Examiner respectfully submits that Schwabe clearly discloses

"loading classes corresponding to the matching class names" *(see Figure 2; Column 4: 1-10, "In*

the JVM, the loading step retrieves the class file representing the desired class."; Column 5: 23-

27, "When the binary file 60 is referenced by an application executing on a virtual machine 65,

a loader 70 loads the binary file 60. A verifier 75 verifies the binary file 60 at some point prior to

execution by an interpreter 80."). Note that, as clarified hereinabove, the verification process

occurs after a class file is loaded. Thus, after matching the set of classes and interfaces between

the old API definition file and the new API definition file occurs, the matching set of classes and

interfaces are loaded to be verified by the verifier.

      Therefore, for at least the reasons set forth above, the rejections made under 35 U.S.C. §

103(a) with respect to Claims 1, 10, and 19 are proper and therefore, maintained.

Conclusion

14.   **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time

policy as set forth in 37 CFR 1.136(a).

      A shortened statutory period for reply to this final action is set to expire THREE

MONTHS from the mailing date of this action. In the event a first reply is filed within TWO

MONTHS of the mailing date of this final action and the advisory action is not mailed until after

the end of the THREE-MONTH shortened statutory period, then the shortened statutory period

will expire on the date the advisory action is mailed, and any extension fee pursuant to 37

CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event,

however, will the statutory period for reply expire later than SIX MONTHS from the mailing

date of this final action.

15.   Any inquiry concerning this communication or earlier communications from the

Examiner should be directed to Qing Chen whose telephone number is 571-270-1071. The

Examiner can normally be reached on Monday through Thursday from 7:30 AM to 4:00 PM.

The Examiner can also be reached on alternate Fridays.

        If attempts to reach the Examiner by telephone are unsuccessful, the Examiner's

supervisor, Wei Zhen, can be reached on 571-272-3708. The fax phone number for the

organization where this application or proceeding is assigned is 571-273-8300.

        Any inquiry of a general nature or relating to the status of this application or proceeding

should be directed to the TC 2100 Group receptionist whose telephone number is 571-272-2100.

        Information regarding the status of an application may be obtained from the Patent

Application Information Retrieval (PAIR) system. Status information for published applications

may be obtained from either Private PAIR or Public PAIR. Status information for unpublished

applications is available through Private PAIR only. For more information about the PAIR

system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR

system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Q. C./

Examiner, Art Unit 2191

/Wei Y Zhen/

Supervisory Patent Examiner, Art Unit 2191